Remote Pre-advising System Using Cell Phone

Prepared By:

Raisa Nazrana Noor

Student ID: 09201025

Department of Computer Science and Engineering

BRAC University

Supervised By:

Abu Mohammad Hammad Ali

Lecturer

Department of Computer Science and Engineering

BRAC University

December, 2011

Declaration

I hereby declare that this thesis is based on the results found by myself. Materials of work found by other resources are mentioned by reference.

_____	_____
Signature of Supervisor	Signature of Author

ACKNOWLEDGMENTS

Special thanks to Abu Mohammad Hammad Ali Sir for being patient and considerate with me and helping me in the development of this work.

Abstract

The proposed system will be able to perform authenticated remote pre-advising of BRAC university students. This is a user friendly system which will enable BRAC University students to register for the courses they want to take in their upcoming semester. Currently students can complete their pre-advising from inside or outside of the university entering into BRAC university website. They even have remote access which requires internet connection. The proposed system is an alternative pre-advising system where students can register for courses by sending SMS (Short Message Service) from their cell phones. This is a handy process for completing pre-advising in a short time. This system maintains the security issues of the students as it has authentication process. The proposed system is not a replacement for the current methods. It is an alternative in the absence of current methods.

TABLE OF CONTENT

1. Introduction

In this thesis, a remote pre-advising system is tried to be developed where students of BRAC University will be able to register for the courses they are willing to take in their upcoming semester from their cell phone. This doesn't require any internet connection. They will have to send a SMS from their cell phone to a certain mobile phone no provided by BRAC University. The SMS will contain the information regarding the courses they want to take. BRAC university server will receive the SMSs and process them to update the main database. Thus the students complete their pre-advising remotely. For this thesis, a prototype of BRAC university server is created.

1.1 Pre-advising:

In BRAC University, students undergo a pre-advising procedure every semester when they register for the courses they want to take in the upcoming semester. From the offered course of the specific semester, students choose the courses they want and book them. Each student is allotted a maximum number of courses to register in a semester, based on his current CGPA. The students then need to wait for the assigned advising day in order to confirm the courses with his advisor and complete his registration for the selected courses.

1.2 SMS:

Short Message Service (SMS) is the text communication service component of phone, web, or mobile communication systems, using standardized communications protocols that allow the exchange of short text messages between fixed line or mobile phone devices.SMS is a part of the **Global System for Mobile Communications** (GSM) as a means of sending messages to and from GSM mobile handsets.

1.3 Authentication:

Authentication is the act of establishing or confirming something (or someone) as authentic. In this thesis, authentication refers to **access control**. The system is supposed to be used only by authorized people and it will detect and exclude the unauthorized. Access to it is therefore controlled by insisting on an authentication procedure to establish with the identity of the user. In this system, students' preset password against their IDs will be the key to authentication.

1.4 Prototype:

A prototype is an early sample or model built to test a concept or process or to act as a thing to be replicated or learned from. For pre-advising of BRAC university students, it is needed to access BRAC University Student Database for accomplishing the pre-advising process. As it is not allowed to access BRAC University's database due to security reasons, a prototype as in a model of BRAC university database is created to perform this thesis work. A normal computer will serve as the BRAC university server and a random mobile number will represent the mobile number students are supposed to send SMS to.

2. Background

For developing a new system, it is needed to know the current methods of pre-advising system that students use for their pre-advising and the problem they face while going through those methods.

2.1 Existing system

2.1.1 Process

The current pre-advising system is entirely web-based. The pre-advising can be done from BRAC University or a student can do it from outside i.e. his home. A student logs into the

university website (www.BRACuniversity.ac.bd/ro) and is asked to enter his/her ID and password to log in to the system. Without logging in, the user can see class schedules, exam schedules, list of pre-requisite courses and seat status of the offered courses. When the student logs in to the system he/she is shown their grade sheet. While logged in, he can go to add course where he can browse through a list of courses offered in that semester. Each student is allotted a maximum number of courses to register in a semester, based on his current CGPA.

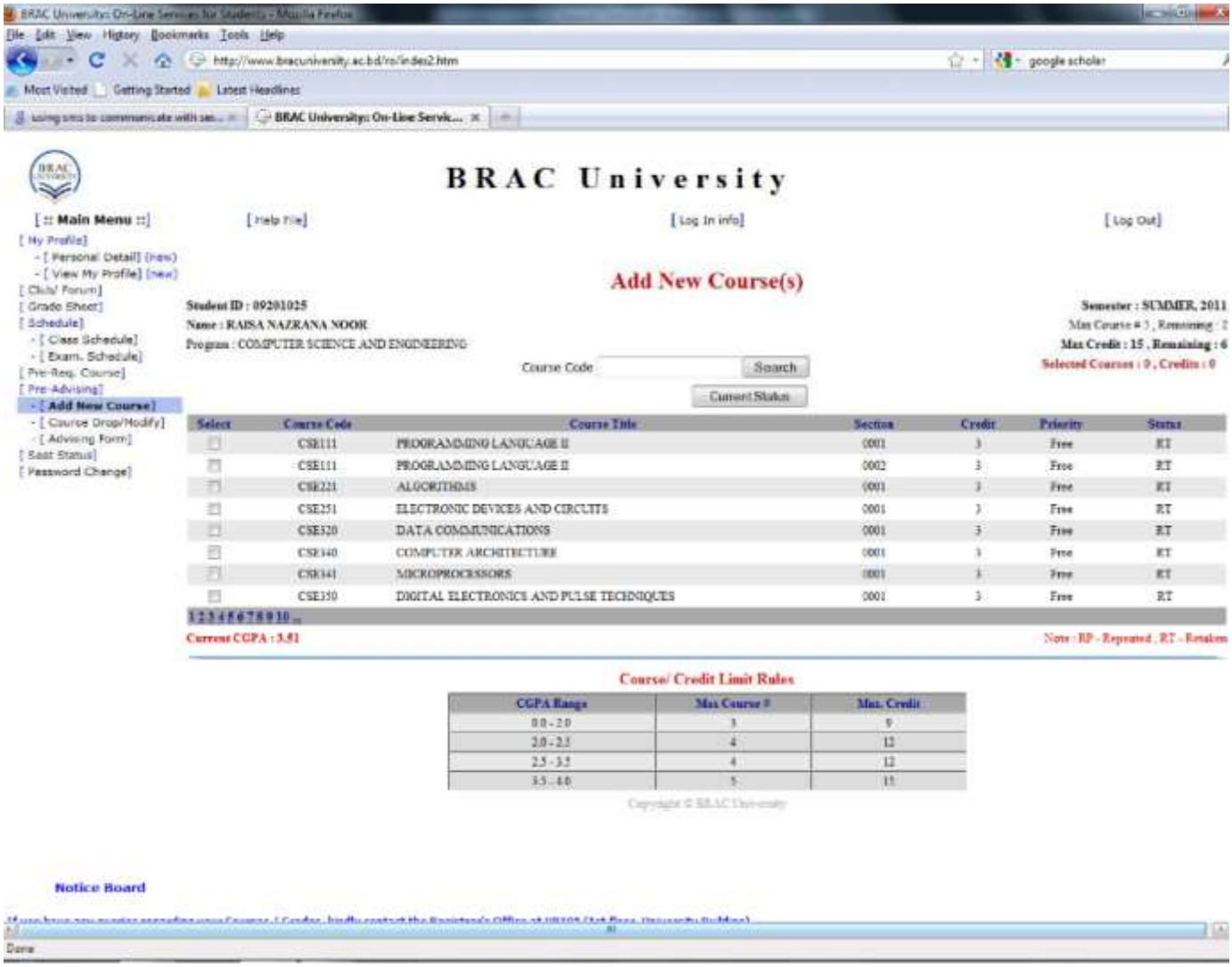

Fig: Web-based pre-advising (course add)

From the list, he can search for the courses he wants to take and adds the courses. Then he can view current status if there are any clashes between courses. When he is satisfied with the course selection, he saves save the changes and the courses are booked for him. He can then view the advising form for the next semester. A student can then drop courses and adds new courses following the same process.

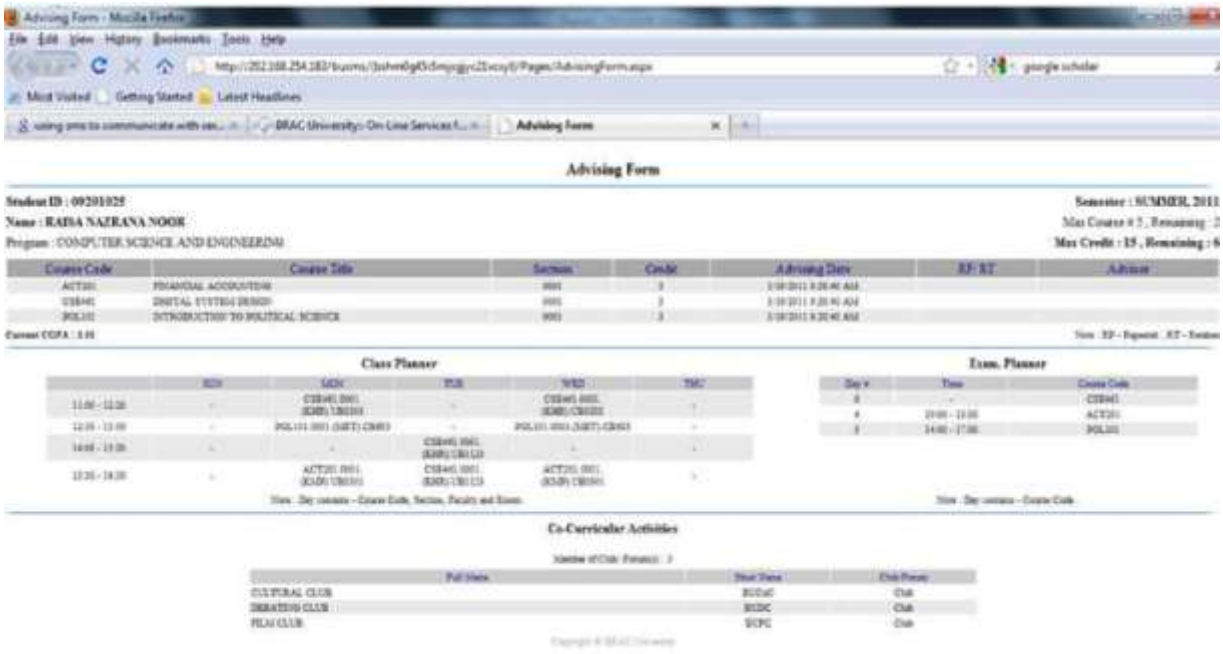

Fig: BRAC University Advising Form

When a student completes his pre-advising, the main database is updated according to the changes the student makes.

- Server identifies a student by his ID and password. Thus it ensures the security of the students' information.
- The server also maintains the system where student can only register for a specific number of courses in accordance with his CGPA.
- The server ensures that a student can't take a course without completing its pre-requisite courses.

2.1.2 Time Restriction

Students are given a time restriction for their pre-advising. The time limit is of 30mins according to the credits completed by the students. The date and time for pre-advising is announced beforehand.

For example, if a student has 90+ credits completed, s/he is assigned a time limit from 9 am to 9:30am on the day of pre-advising. If a student fails to complete his/her pre-advising within the given time limit, s/he can't undergo the pre-advising. He can't add any new course after that time.

2.2 Problem with existing system

When students do their pre-advising in the existing process, they often face some problems which sometimes lead to an unsuccessful pre-advising.

- Students are allotted a small specific time to complete their pre- advising. If they fail to log in to the BRAC university website on time they can't do a pre-advising at all.
- As many of the students attempt to do their pre-advising the server becomes slow and pre-advising becomes time consuming.
- Sometimes log in process creates problem and it gives error when students try to log in. Sometimes students can't log in at all.
- If a student wants to do pre-advising from home, and in case there is no internet connection at that moment, the student faces the risk of no pre-advising.

3. Proposed System

The proposed system will be able to perform remote pre-advising by sending SMS from a cell phone. This system is not a replacement of the current method which is online pre-advising. It is an alternative. When a student fails to do the pre-advising in the usual way, he can send a SMS to perform the pre-advising easily from anywhere with a mobile network coverage.

A mobile application will be saved in the cell phone of the students. They will fill up the form and send the SMS. The SMS will be sent to the receiver device by the cellular network. When server sends a reply SMS in accordance with the sent SMS, the sender mobile device will receive the reply via cellular network.

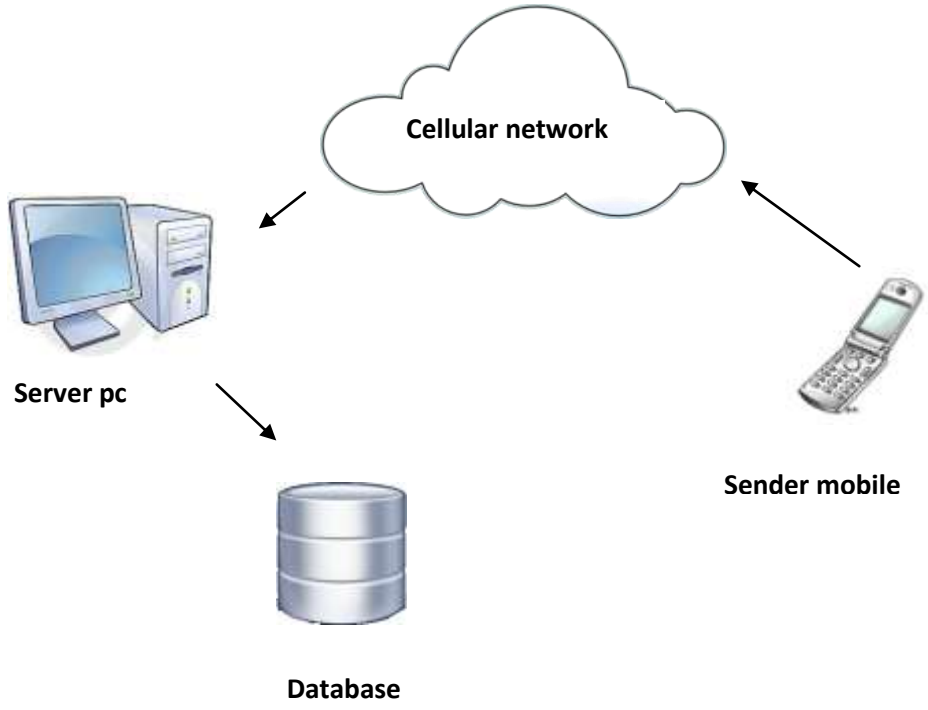

Fig: System architecture

3.1 Portals

There are two portals in this remote pre-advising system. The one which is used by the students is the User end and another is for the official use which is the server end.

3.1.1 User End

The Function of the user end is to send information for the pre-advising via SMS. In the user end, students will save a mobile application in their cell phone. The application contains of a form. Students will fill up the form with the required information. When they send SMS, the application generates a SMS using the information given in the form and sends the SMS. The cellular network is responsible for the delivery of the SMS. The student will then receive a SMS with the information of success or failure of the pre-advising.

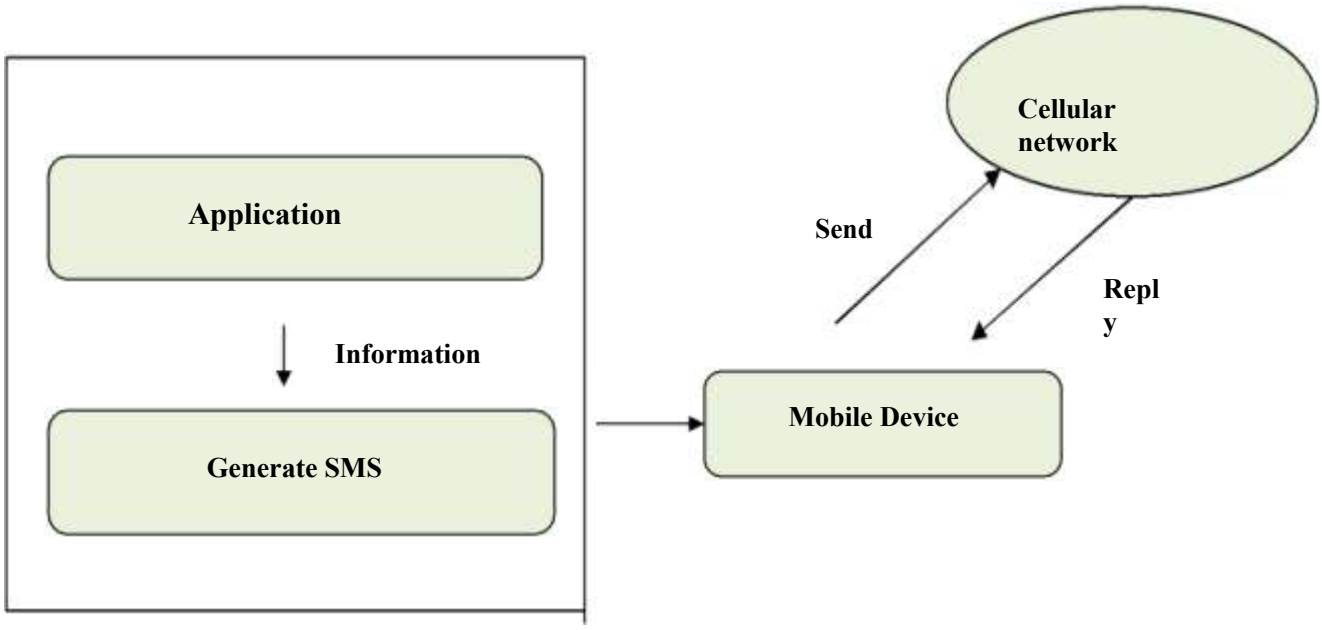

Fig: User end

Mobile Application instead of Normal Text Message

There are several reasons behind using a mobile application instead of writing a simple SMS
from the write message option of a cell phone.

- The application works as a guideline for the students to give all the required

information for this remote pre-advising system. If they had to write a normal SMS, there is a possibility that they may not give important information such as password or section number or they may forget to give space between information. This would have lead to an unsuccessful pre-advising.

- The application checks for error so that students can avoid some technical mistakes such as inserting an incomplete Student ID. If a student had to write a normal SMS, it would have been difficult for him to maintain the right order of the information. As there is much information to give, it is very much possible to swap the positions of the information. For example: student needs to give the course name and then section number of that course. If he mistakenly gives section number before the course name it will lead to an unsuccessful pre-advising.
- Using Timestamp, students can specify the time to deliver their SMS. So they can send the SMS in advance to avoid the risk of missing the due time slot of pre-advising.

3.1.2 Server End

Server end basically receives a SMS for pre-advising, process that SMS, searches the database and sends a reply of the action taken after all the procedure. There are three parts in the server end.

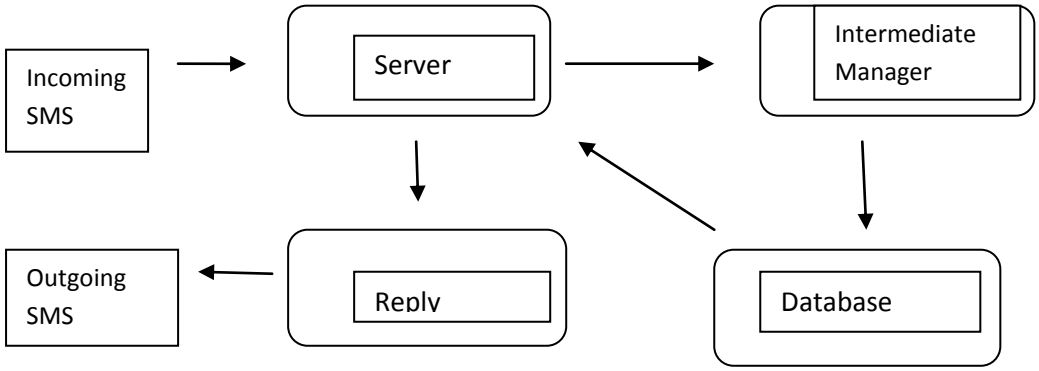

Fig: Server End

3.1.2.1 Server

The test message is sent to and received by the server. There is a java application which extracts the information from the received SMS by parsing the text, searches the main database, updates database is all the necessary information given are accurate and sends reply message to the sender for the error or confirmation of pre-advising.

3.1.2.2 Database

Database is the main database where all the information regarding pre-advising process is kept. Server accesses the database with intermediate manager and the information is searched and updates accordingly.

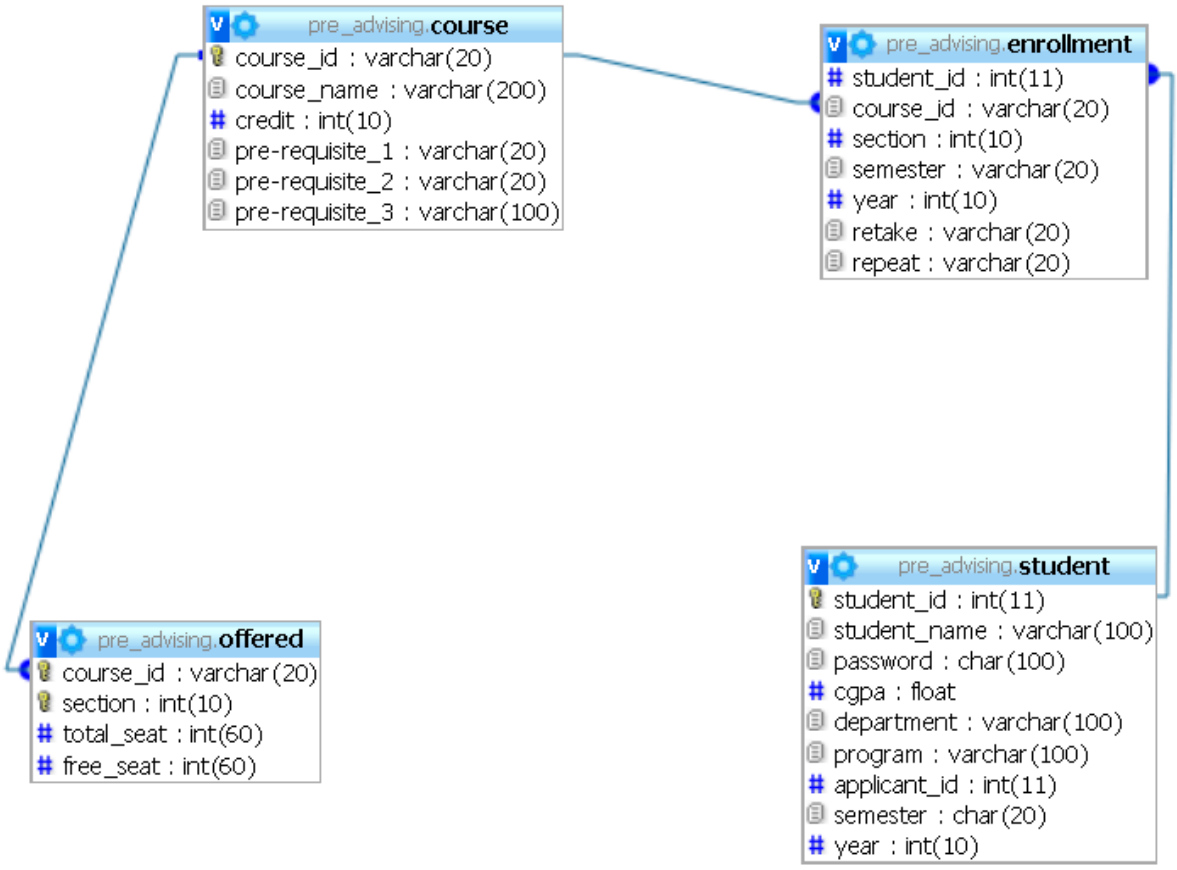

Fig: Class diagram of Database

3.2 Procedure

3.2.1 Creating SMS

When a student is done with filling up the application form, a SMS is created using the information given. The architecture includes three layers: Wireless Messaging API, Implementation layer and low level transport layer.

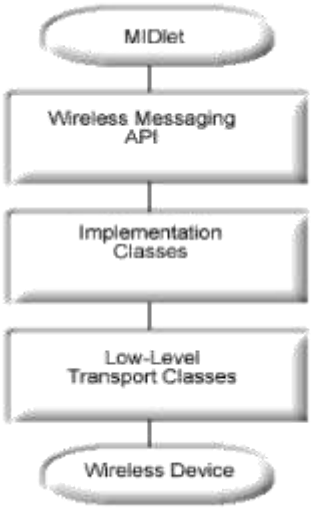

Fig: Architecture of creating SMS

3.2.2 Process SMS in Server

At the server end, when a SMS is received, The SMS is then transferred to the server pc. Then the SMS is processed and the information required for pre-advising is extracted for the use of the java application which acts as the intermediate manager between the three parts of the server end.

3.2.3 Database Access

The java application then makes a connection with the main database using JDBC (Java Database Connectivity) API. With the information extracted from the incoming SMS, the database id searched and updated if necessary.

- The database first searches for the Student ID and matches the password. If both are correct, the system proceeds to the next step. If the ID or password is invalid, the

process stops and a SMS is sent to notify the sender about it.

- If a student attempts for more course than his maximum limit, the database is not updated and a student is notified with the reason.
- The database checks if a student requests for a course without completing its pre-requisite course, a notification is sent to the student.
- If a student wants to drop a course which he hadn't added then he is notified about it.
- If all the seats are booked of a course, the database is not updated and the student is notified.
- If all the information are correct and there is no problem regarding database, a student's pre-advising is allowed. The database is updated and the student receives a confirmation message.

3.2.4 Sending Notification SMS

Whenever there is a mismatch while searching the database or whenever the search doesn't return any result, it is considered that there is a problem with the information given by the student when the filled-up the application form in his mobile or it is not possible to update the database due to some specific reason. The student is notified with the failure notification accompanied with the reason (if the reason is known).

When the pre-advising process is successful and the database is updated and the specific course and section are booked against a student's ID, the student is sent a confirmation message.

3.3 Advantage of the proposed System

The proposed system has certain advantages. That's why it can be an appropriate alternative of the current pre-advising system.

- Requires no internet connection. An SMS will be enough to register for wanted courses.
- Easy and simple solution of pre-advising. A student needs to fill-up a form and send the SMS. No need to go through several steps to add courses.

- SMS can be sent from any place with mobile network coverage.

- Takes less time to complete the pre-advising at user end. As the students need to fill up the form, they don't have to wait for loading anything else like web-based system.
- No risk of missing the fixed time slot of pre-advising if time for sending the message is set in advance.

4. Implemented System

The proposed System could not be implemented entirely because of technical and other challenges. So in the replacement of the proposed system, a different approach was used where at the user end, a mobile application has been created to send the SMS with necessary information for pre-advising. At the server end a GUI application was developed to represent the server end with database connection.

4.1 User End

4.1.1 Application form

This is a user-end process where a student opens the mobile application and fill-up the form. He gives his Student ID, password, course IDs and sections for respective courses and chooses an option to add or drop a course. Each student can add at most 5 courses. Then he proceeds to the next step where he has to insert phone no provided by the University (in case of this thesis, the number is a specific number determined by the thesis member). When he presses the send button, a SMS is created and sent.

Fig: Form fill-up

4.1.2 Error checking

The application form is used as a system which finds the common error and notifies students to modify the errors before sending the SMS. As the common errors are checked beforehand, it the process faces less problem while searching and updating the database in server end. Though this process doesn't solve all the errors regarding pre- advising, it checks for the common ones.

- If a student doesn't give student ID or password or doesn't give at least one course and section, he is notified.

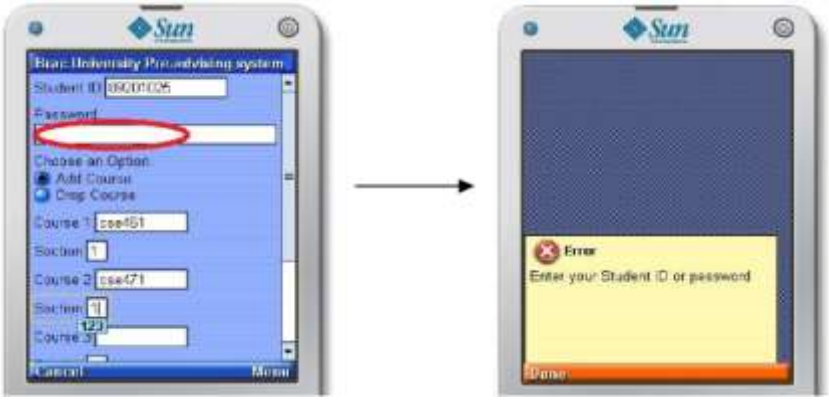

Fig: Incomplete form

- If a student gives invalid information, he is notified. For example, student IDs are of 8 digits in BRAC University. If a student gives and ID which is less than 8, he will be notified to recheck. In another case, course IDs are of length 6. If a student gives a course id which is of length less than 6, he will be notified.

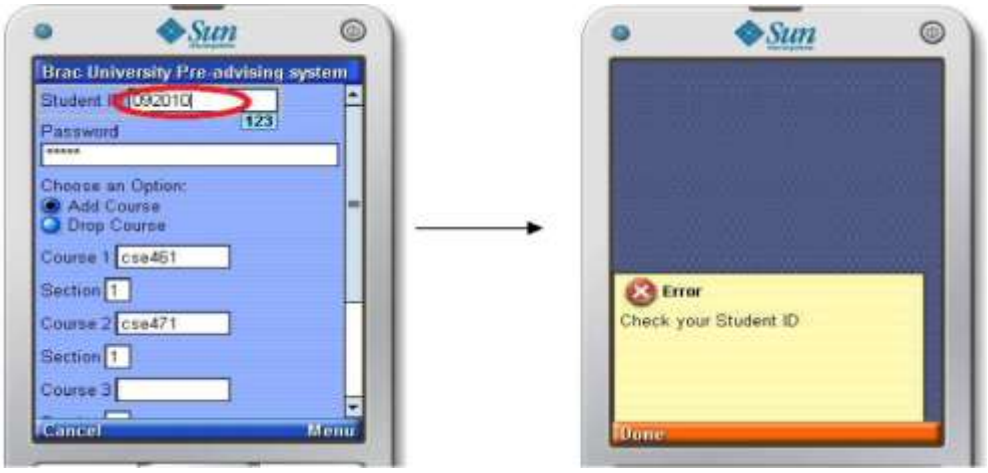

Fig: Invalid Student Id

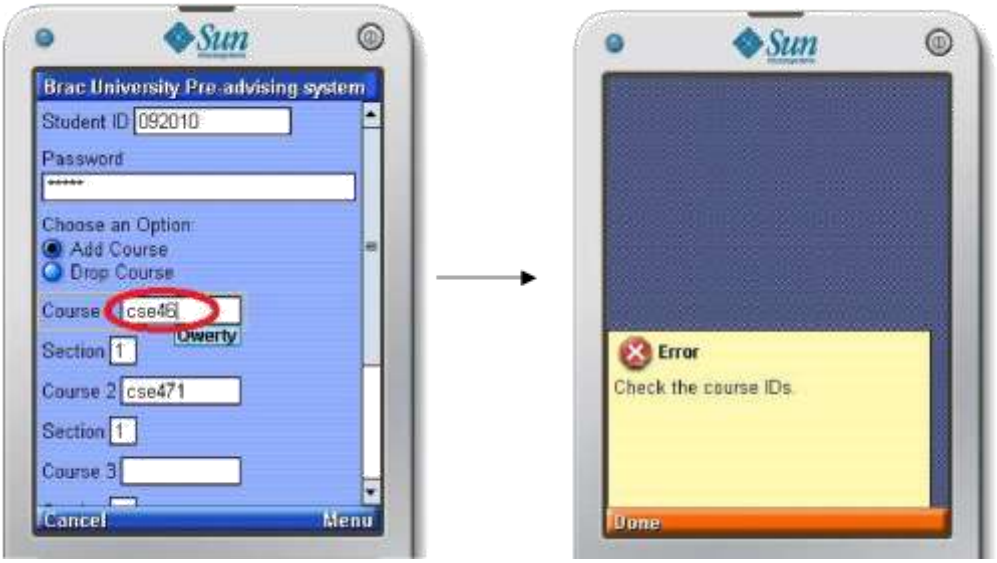

Fig: Invalid Course ID

4.2 Server End:

A GUI application has been developed to represent the services at the server end. There are three steps to represent the procedure.

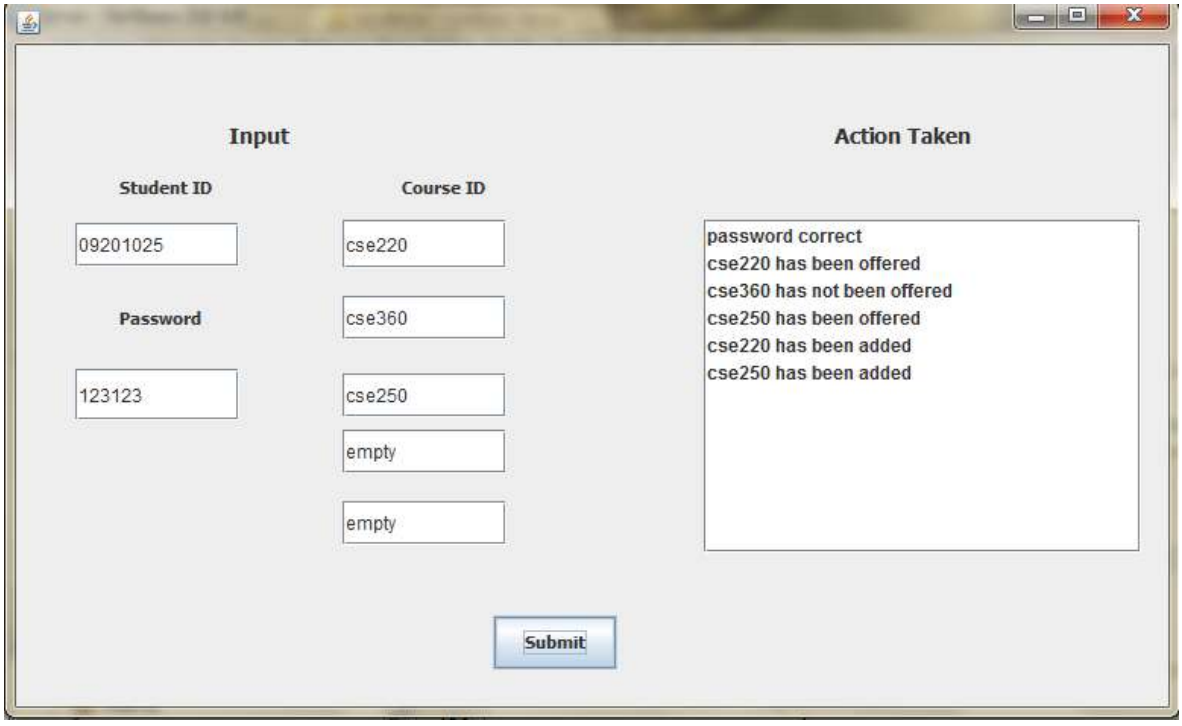

Fig: GUI view

- **Input**

  The necessary information for a pre-advising request is given in the input panel to search the database. Input errors are handled. It represents the information received from a text message in the proposed system.

- **Database Access**

  With the given information in the input panel, the database is searched for password checking, pre-requisite course checking, offered courses etc. If the necessary information is correct then the database is updated with courses taken for pre advising.

- **Output**

  The results of database search and updates are shown in the output panel. If there is any error as in incorrect password, information of requested courses being not offered, the message is shown in the output panel. If information is correct and database is updated, the notification is also shown in the output panel. The represents the notification message a server sends to the user.

5. Challenges

While developing the system, several challenges were faced.

- The system was tried to be developed using Servlet in Java EE platform. But for technical problem regarding parsing SMS, connection with glassfish and time management, the approach was dropped and simple java application was built.
- Creation of server using java failed. The server could not receive text messages. Technical issues and developer's limitation can be named for the cause behind this. A GUI application was created to represent the actions of the server.
- At user end, text message is sent and a notification confirms the sending process. But that SMS can't be received. No particular reason could be found. Different approaches were taken to solve the problem. But till now, no solution could be found.

6. Technology

Device:

- Properly functioning mobile phones for both user and server end.
- Desktop or laptop PC.

Platform:

- For mobile device, J2ME (Java 2 Micro Edition) platform is used
- For PC environment, Java platform is used.

Software:

- Java(TM) ME platform SDK 3.0
- Netbeans 6.9.1

Database: For database, MySQL is used.

Connection:

- The connection between the user end and server end will be handled by existing cellular network.

7. Installation:

A student needs to save the executable JAR file in his cell phone and open it from the mobile and the application will be ready to use.

8. Assumptions

For developing the system some facts are assumed.

- Students have properly functioning mobile handset which will support the mobile application used for the pre-advising system.

- Students have an idea about the courses they are willing to take in the next semester.

9. Limitations

There are some limitations in this system.

- This system is only applicable for courses offered by Computer Science and Engineering Department of BRAC University. The students can only register for CSE courses in this pre-advising system.

- Students have limited options. The can only add/drop courses and choose section. But they cannot view seat status, offered course list, schedule or pre-requisite course list.

- Students need to have a cell phone that will support the mobile application.

10. Risk factor

The thesis process encountered various risks at any moment.

Technical risks:

- The Database crashed without warning during the development of the system. To recover from this disaster, backup was maintained.

People risks:

- While working on the thesis project, team member got ill. As there is only one member doing this specific thesis, thesis came to a stop until the member recovered from the illness.

Estimation risks:

- There is a risk that thesis member failed to reach the final goal of the thesis within the given timeline.

11. Expandability

The thesis has the scope to expand the proposed system. In future it is possible to modify the system to add new features to make this remote pre-advising as a replacement of the current method. Some of the scopes can be:

- Students can send SMS asking for the course list which will be offered in the next semester before attempting pre-advising via SMS. The BRAC university server will send a list of courses by a reply SMS.
- Students can ask for the sections and seat status of a specific course before attempting pre-advising via SMS. The BRAC university server will send the asked information by a reply SMS.
- Students can ask for the pre-requisite list for one or more courses before attempting to register for the courses.

12. Future Plan

With some more time, some development can be done with the project. It will be tried to solve the existing problems the current system such as receiving SMS. A server will be created with another approach and make it work and make connection between user and server end.

References

- Li, Sing & Knudsenli, Jonathan Beginning, *J2ME: From Novice to Professional*, Third Edition

- O'reilly, *Learning wireless Java*, ISBN 0-59600-243-2,2001

- Study of SMS Security as part of an electronic voting system by Chowdhury Mushfiqur Rahman &Shah Md. Adnan Khan

- Design and implementation of BRAC university online advising application by Hasan Ferdouse Sharif, Nadeem Hussai, Saniun Saber Chowdhury

- http://www.w3schools.com/

- https://www.aloashbei.com.bd/wiki/api-specifications

- http://www.java2s.com/Code/Java/J2ME/ExampleMIDlet.htm

- http://www.tutorialspoint.com

- http://www.developershome.com/

Codes

User end:

```
import java.io.IOException;
import java.io.InterruptedIOException;
import java.util.Calendar;
import java.util.Timer;
import javax.microedition.io.ConnectionNotFoundException;
import javax.microedition.io.Connector;
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.ChoiceGroup;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.Form;
import javax.microedition.lcdui.TextField;
import javax.microedition.midlet.*;
import javax.microedition.sensor.Condition;
import javax.wireless.messaging.MessageConnection;
import javax.wireless.messaging.TextMessage;
import java.util.*;
import javax.microedition.lcdui.*;


/**
 * @author Raisa
 */
public class thesis extends MIDlet implements CommandListener, Runnable {

    private Display display;
    private TextField id, password, course1, course2, course3, course4, course5, hour ,min,
date,month,year;
    private Command send, next, cancel, back,exit;
    private Form form1, form2, form3, form4;
    private ChoiceGroup option;
    private String s_id, pass, action, c1, c2, c3, c4, c5,msg;
    private Alert error1,error2,error3,ok;
    MessageConnection conn = null;
    Calendar c= Calendar.getInstance();
    String hr,mn,dt,mnth,yr;


    //private String input = "";
    //private Alert al;
```

```
public thesis() {

    display = Display.getDisplay(this);
    cancel = new Command("Cancel", Command.EXIT, 1);
    next = new Command("NEXT", Command.OK, 1);
    send = new Command("Send", Command.OK, 1);
    back = new Command("Back", Command.BACK, 1);
    exit = new Command("Exit", Command.EXIT, 1);
    course1 = new TextField("Course 1", "", 6, TextField.ANY);
    course2 = new TextField("Course 2", "", 6, TextField.ANY);
    course3 = new TextField("Course 3", "", 6, TextField.ANY);
    course4 = new TextField("Course 4", "", 6, TextField.ANY);
    course5 = new TextField("Course 5", "", 6, TextField.ANY);
    id = new TextField("Student ID", "", 8, TextField.NUMERIC);
    password = new TextField("Password", "", 30, TextField.PASSWORD);
    hour = new TextField("Give a hour from 9-17 ", "", 5, TextField.ANY);
    min = new TextField("Give minute from 00-60 ", "", 5, TextField.ANY);
    date = new TextField("Give a date from 1-31 ", "", 5, TextField.ANY);
    month = new TextField("Give a month from 1-12 ", "", 5, TextField.ANY);
    year = new TextField("Give a year.i.e 2012 ", "", 5, TextField.ANY);
    option = new ChoiceGroup("Choose an Option:", ChoiceGroup.EXCLUSIVE);



    form1 = new Form("Brac University Pre-advising system");



}

public void startApp() {

    option.append("Add Course", null);
    option.append("Drop Course", null);


    form1.addCommand(cancel);
    form1.addCommand(next);
    form1.setCommandListener(this);
    form1.append(id);
    form1.append(password);
    form1.append(option);
    form1.append(course1);
    form1.append(course2);
    form1.append(course3);
    form1.append(course4);
    form1.append(course5);
```

```
    display.setCurrent(form1);

    /*form2.addCommand(next);
    form2.addCommand(cancel);
    form2.setCommandListener(this);
    form2.append(course1);
    form2.append(course2);
    form2.append(course3);
    form2.append(course4);
    form2.append(course5);

    form3.addCommand(next);
    form3.addCommand(cancel);
    form3.setCommandListener(this);
    form3.append(course1);
    form3.append(course2);
    form3.append(course3);
    form3.append(course4);
    form3.append(course5);*/

    form2=new Form("Add a phone No. for receiver");
    form2.addCommand(send);
    form2.addCommand(cancel);
    form2.addCommand(back);
    form2.setCommandListener(this);
    form2.append(hour);
    form2.append(min);
    form2.append(date);
    form2.append(month);
    form2.append(year);

    form3=new Form("");
    form3.addCommand(exit);
    form3.setCommandListener(this);




}

public void pauseApp() {
}

public void destroyApp(boolean unconditional) {
}

public void commandAction(Command c, Displayable d) {

    if (c == cancel) {
```

```
            destroyApp(false);
            notifyDestroyed();
        }

        else if (c == next && d == form1) {

            s_id = id.getString();
            pass = password.getString();
             c1=course1.getString();
             System.out.println(c1);
            c2=course2.getString();
            //System.out.println(c2);
            c3=course3.getString();
            c4=course4.getString();
            c5=course5.getString();
            int length = id.getString().length();

            if (option.getSelectedIndex() == 0) {

                action = "add";
            }
            else if (option.getSelectedIndex() == 1) {
                action = "drop";
            }

            if (id.getString().length()==0 || password.getString().length()==0){

                error1= new Alert("Error", "Enter your Student ID or password", null,
AlertType.ERROR);

            display.setCurrent(error1,form1);
            }

            else if (length<8){

            error2=new Alert("Error", "Check your Student ID", null, AlertType.ERROR);
             display.setCurrent(error2,form1);
            }


            else                                                                    if
(c1.equals("")==true&&c2.equals("")==true&&c3.equals("")==true&&c4.equals("")==true&
&c5.equals("")==true){

                error3= new Alert("Error", "Insert atleast one course id.", null, AlertType.ERROR);
            display.setCurrent(error3,form1);
            }

            else                                                                    if
```

```
((course1.getString().length()!=0&&course1.getString().length()<6)||(course2.getString().lengt
h()!=0&&course2.getString().length()<6)||(course3.getString().length()!=0&&course3.getStrin
g().length()<6)||(course4.getString().length()!=0&&course4.getString().length()<6)||(course5.g
etString().length()!=0&&course5.getString().length()<6)){

            error3= new Alert("Error", "Check the course IDs.", null, AlertType.ERROR);
          display.setCurrent(error3,form1);
          }

          else{display.setCurrent(form2);}

       }

       else if(c==back && d==form2){

          display.setCurrent(form1);
       }

       else if (c==send && d==form2){


        //}

          }

       //else if (c==exit&&d==form3){
          //destroyApp(false);
          //notifyDestroyed();

      // }

       }
   public void sending() {
       msg=s_id+" "+pass+" "+action+" "+c1+" "+c2+" "+c3+" "+c4+" "+c5;
new Thread(this).start();
       //MessageConnection conn = null;
//       try{
//       conn =(MessageConnection) Connector.open("sms://"+"+8801676008312"+":8080");
//       }catch(ConnectionNotFoundException cnf){
//       error1=new Alert("Error", "connection not found", null, AlertType.ERROR);
//          display.setCurrent(error1);
//       }
//
//       catch(IOException cnf){
//
//       }
//
//TextMessage txtmessage = (TextMessage)
conn.newMessage(MessageConnection.TEXT_MESSAGE);
```

```
//txtmessage.setPayloadText(msg);
//        try {
//            conn.send(txtmessage);
//        } catch (IOException ex) {
//         error1=new Alert("Error", "IO exception", null, AlertType.ERROR);
//            display.setCurrent(error1);
//        }
//                    ok= new Alert("Successful", "Your SMS has been sent.", null,
AlertType.CONFIRMATION);
//               display.setCurrent(ok,form3);
//               destroyApp(false);
//               notifyDestroyed();
//

    }


    public void run() {

while(conn==null){
           try {
              conn                                =                                (MessageConnection)
Connector.open("sms://"+"+8801671034525"+":7424");
           } catch (IOException ex) {
              ex.printStackTrace();
           }

        }
        try {

           TextMessage                txtmessage                =                (TextMessage)
conn.newMessage(MessageConnection.TEXT_MESSAGE);
txtmessage.setPayloadText(msg);

           conn.send(txtmessage);
        } catch (IOException ex) {
           ex.printStackTrace();
        }
 ok=       new       Alert("Successful",       "Your       SMS       has       been       sent.",       null,
AlertType.CONFIRMATION);
              display.setCurrent(ok,form3);
              destroyApp(false);
              notifyDestroyed();
    }
}
Server end:


import com.mysql.jdbc.MySQLConnection;
```

```
import com.mysql.jdbc.Statement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;


/**
 * @author Raisa
 */
public class Receiver extends javax.swing.JFrame {



    static Connection conn;
    static String studentID = "empty";
    static String password = "empty";
    static String section = "1";
    static String course1 = "empty";
    static String course2 = "empty";
    static String course3 = "empty";
    static String course4 = "empty";
    static String course5 = "empty";
    static String semester = "Spring";
    static String year = "2012";
    static String[] data= new String[10];
    static int i=0;

    static String passCheck, offer1, offer2, offer3, offer4, offer5, preReqCheck1,
preReqCheck2, preReqCheck3, taken;
    static boolean pss, ofr1,ofr2,ofr3,ofr4,ofr5, preReq1,preReq2,preReq3, tkn;
    static int count;
    public Receiver() {
        initComponents();

        jList1.setListData(data);
        data[i]= "your output here";


jList1.updateUI();

    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jTextField6 = new javax.swing.JTextField();
        jPanel1 = new javax.swing.JPanel();
```

```
jLabel1 = new javax.swing.JLabel();
jTextField1 = new javax.swing.JTextField();
jLabel2 = new javax.swing.JLabel();
jLabel3 = new javax.swing.JLabel();
jLabel5 = new javax.swing.JLabel();
jTextField5 = new javax.swing.JTextField();
jTextField7 = new javax.swing.JTextField();
jTextField8 = new javax.swing.JTextField();
jTextField9 = new javax.swing.JTextField();
jLabel6 = new javax.swing.JLabel();
jButton1 = new javax.swing.JButton();
jTextField2 = new javax.swing.JTextField();
jScrollPane1 = new javax.swing.JScrollPane();
jList1 = new javax.swing.JList();
jTextField3 = new javax.swing.JTextField();

jTextField6.setText("jTextField6");

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 14));
jLabel1.setText("Input ");

jTextField1.setText("09201025");
jTextField1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField1ActionPerformed(evt);
    }
});

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel2.setText("Student ID");

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel3.setText("Password");

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 11));
jLabel5.setText("Course ID");

jTextField5.setText("empty");
jTextField5.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField5ActionPerformed(evt);
    }
});

jTextField7.setText("empty");
jTextField7.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jTextField7ActionPerformed(evt);
    }
});

jTextField8.setText("empty");
jTextField8.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField8ActionPerformed(evt);
    }
});

jTextField9.setText("empty");
jTextField9.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField9ActionPerformed(evt);
    }
});

jLabel6.setFont(new java.awt.Font("Tahoma", 1, 14));
jLabel6.setText("Action Taken");

jButton1.setFont(new java.awt.Font("Tahoma", 1, 11));
jButton1.setText("Submit");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

jTextField2.setText("123123");
jTextField2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField2ActionPerformed(evt);
    }
});

jScrollPane1.setViewportView(jList1);

jTextField3.setText("empty");
jTextField3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTextField3ActionPerformed(evt);
    }
});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
        .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(128, 128, 128)
                    .addComponent(jLabel1,   javax.swing.GroupLayout.PREFERRED_SIZE,   79,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(302, 302, 302)
                    .addComponent(jButton1,  javax.swing.GroupLayout.PREFERRED_SIZE,  82,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(27, 27, 27)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                        .addComponent(jTextField2,     javax.swing.GroupLayout.DEFAULT_SIZE,
108, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G, false)
                            .addComponent(jLabel2,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    79,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jTextField1,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 108, Short.MAX_VALUE)
                            .addComponent(jLabel3,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    79,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                    .addGap(25, 25, 25)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(82, 82, 82)
                            .addComponent(jLabel5,   javax.swing.GroupLayout.PREFERRED_SIZE,
55, javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGroup(jPanel1Layout.createSequentialGroup()
                            .addGap(43, 43, 43)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                                .addComponent(jTextField3,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 108, Short.MAX_VALUE)
```

```
                        .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(jTextField5,
javax.swing.GroupLayout.DEFAULT_SIZE, 108, Short.MAX_VALUE)
                        .addComponent(jTextField7,
javax.swing.GroupLayout.DEFAULT_SIZE, 108, Short.MAX_VALUE)
                        .addComponent(jTextField9,
javax.swing.GroupLayout.DEFAULT_SIZE, 108, Short.MAX_VALUE)
                        .addComponent(jTextField8,
javax.swing.GroupLayout.DEFAULT_SIZE, 108, Short.MAX_VALUE)))))
                .addGap(124, 124, 124)))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILI
NG)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jLabel6)
                    .addGap(124, 124, 124))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jScrollPane1,   javax.swing.GroupLayout.PREFERRED_SIZE,
288, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addContainerGap())))
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(33, 33, 33)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(jLabel6,     javax.swing.GroupLayout.PREFERRED_SIZE,     22,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel1,     javax.swing.GroupLayout.PREFERRED_SIZE,     29,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                .addComponent(jLabel2,     javax.swing.GroupLayout.PREFERRED_SIZE,     29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel5,     javax.swing.GroupLayout.PREFERRED_SIZE,     22,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
                .addGroup(jPanel1Layout.createSequentialGroup()
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                    .addComponent(jTextField1,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField3,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    32,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                    .addComponent(jTextField5,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELI
NE)
                    .addComponent(jTextField7,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jTextField2,
javax.swing.GroupLayout.PREFERRED_SIZE,                                    34,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jTextField9,    javax.swing.GroupLayout.PREFERRED_SIZE,
29, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jTextField8,    javax.swing.GroupLayout.PREFERRED_SIZE,
29, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(46, 46, 46)
                .addComponent(jButton1,  javax.swing.GroupLayout.PREFERRED_SIZE,  36,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addComponent(jScrollPane1,      javax.swing.GroupLayout.PREFERRED_SIZE,
219, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap())
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
       layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
       .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
          .addContainerGap()
          .addComponent(jPanel1,                javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(jPanel1,                javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        //studentID=jTextField1.getText();
        //System.out.println(studentID);
    }

    private void jTextField5ActionPerformed(java.awt.event.ActionEvent evt) {
        //course2=jTextField5.getText();
    }

    private void jTextField7ActionPerformed(java.awt.event.ActionEvent evt) {
         //course3=jTextField7.getText();
    }

    private void jTextField9ActionPerformed(java.awt.event.ActionEvent evt) {
         //course4=jTextField9.getText();
    }

    private void jTextField8ActionPerformed(java.awt.event.ActionEvent evt) {
         //course5=jTextField8.getText();
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

        studentID=jTextField1.getText();
        password=jTextField2.getText();
        course1=jTextField3.getText();
        //System.out.println(course1);
        course2=jTextField5.getText();
        course3=jTextField7.getText();
       course4=jTextField9.getText();
       course5=jTextField8.getText();
        try {
            Class.forName("com.mysql.jdbc.Driver").newInstance();
        } catch (Exception ex) {
```

```
// do nothing
        }
        conn = null;

        try {
            String url = "jdbc:mysql://localhost:3306/pre_advising";
            conn = DriverManager.getConnection(url, "root", "root");
            //System.out.println("yes");

        } catch (SQLException ex) {
// handle any errors
            System.out.println("SQLException: " + ex.getMessage());
            System.out.println("SQLState: " + ex.getSQLState());
            System.out.println("VendorError: " + ex.getErrorCode());
        }


        search();
        insert();
    }

    private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }


    private  void search() {

 //Searching Initiates

        Statement stmt = null;
        ResultSet rs = null;
        try {
            stmt = (Statement) conn.createStatement();
            //System.out.println("yes1");
            //rs = stmt.executeQuery("SELECT password FROM student where student_id =
'"+studentID+"'");

//password verify
            if(studentID.equals("empty")==false&&password.equals("empty")==false){
            if (stmt.execute("SELECT password FROM student where student_id = '" + studentID
+ "')) {
                rs = stmt.getResultSet();
                 //System.out.println(studentID);
```

```
            while (rs.next()) {
              // System.out.println("yes3");
               passCheck = rs.getString("password");
               //System.out.println(password);
               //System.out.println(passCheck);
               if (passCheck.equalsIgnoreCase(password) == true) {
                   System.out.println(passCheck);
                   data[i++]="password correct";
           jList1.updateUI();

                   pss = true;

               }


            else  {
                   data[i++]="password incorrect";
           jList1.updateUI();
               }
            }
    }
       }


          if (pss==true){
//course offering verify for course1


          if (course1.equals("empty")==false) {
             if ((stmt.execute("SELECT course_id FROM offered where course_id = '" + course1
+ "'"))) {
               rs = stmt.getResultSet();


                if(rs != null && rs.next()){
               //while (rs.next()) {
                   offer1 = rs.getString("course_id");

                   System.out.println(offer1);

                   if (offer1.equalsIgnoreCase(course1) == true) {
                       data[i++]=course1+" has been offered";
           jList1.updateUI();
                          ofr1 = true;

                   }
                   //}
               }
```

```
                else  {
                        data[i++]=course1+" has not been offered";
            jList1.updateUI();
                    }
                }
              }

//course offering verify for course2

            if (course2.equals("empty")==false) {
              if ((stmt.execute("SELECT course_id FROM offered where course_id = '" + course2
+ "'"))) {
                rs = stmt.getResultSet();


                 if(rs != null && rs.next()){
                // while (rs.next()) {
                    offer2 = rs.getString("course_id");

                    System.out.println(offer2);

                    if (offer2.equalsIgnoreCase(course2) == true) {
                        data[i++]=course2+" has been offered";
            jList1.updateUI();
                       ofr2 = true;

                    }
                    //}
                }
                else  {
                        data[i++]=course2+" has not been offered";
            jList1.updateUI();
                    }
                }
              }


//course offering verify for course3
          if (course3.equals("empty")==false) {
              if ((stmt.execute("SELECT course_id FROM offered where course_id = '" + course3
+ "'"))) {
                rs = stmt.getResultSet();


                 if(rs != null && rs.next()){
                //while (rs.next()) {
                    offer3 = rs.getString("course_id");

                    if (offer3.equalsIgnoreCase(course3) == true) {
```

```
                    data[i++]=course3+" has been offered";
            jList1.updateUI();
                       ofr3 = true;

                   }
                   //}
               }
               else  {
                    data[i++]=course3+" has not been offered";
            jList1.updateUI();
                   }
               }
             }

//course offering verify for course4

            if (course4.equals("empty")==false) {
              if ((stmt.execute("SELECT course_id FROM offered where course_id = '" + course4
+ "'"))) {
                  rs = stmt.getResultSet();


                   if(rs != null && rs.next()){
                  //while (rs.next()) {
                     offer4 = rs.getString("course_id");

                     if (offer4.equalsIgnoreCase(course4) == true) {
                         data[i++]=course4+" has been offered";
            jList1.updateUI();
                       ofr4 = true;

                     }
                     //}
                  }
                  else  {
                        data[i++]=course4+" has not been offered";
            jList1.updateUI();
                     }
                  }
              }

//course offering verify for course5
           if (course5.equals("empty")==false) {
              if ((stmt.execute("SELECT course_id FROM offered where course_id = '" + course5
+ "'"))) {
                  rs = stmt.getResultSet();


                   if(rs != null && rs.next()){
```

```
            // while (rs.next()) {
                offer5 = rs.getString("course_id");

                if (offer5.equalsIgnoreCase(course5) == true) {
                    data[i++]=course5+" has been offered";
          jList1.updateUI();
                   ofr5 = true;

                }
                //}
            }
            else  {
                    data[i++]=course5+" has not been offered";
          jList1.updateUI();
                }
            }
          }


        }


      } catch (SQLException ex) {
// handle any errors
          System.out.println("SQLException: " + ex.getMessage());
          System.out.println("SQLState: " + ex.getSQLState());
          System.out.println("VendorError: " + ex.getErrorCode());
      } finally {

          if (rs != null) {
             try {
                rs.close();
             }
             catch (SQLException sqlEx) {
             } // ignore
             rs = null;
          }
          if (stmt != null) {
             try {
                stmt.close();
             } catch (SQLException sqlEx) {
             } // ignore
             stmt = null;
          }
      }


   }
```

```
    private  void insert(){

        Statement stmt = null;
        //ResultSet rs = null;
        try {
           stmt = (Statement) conn.createStatement();
    if (pss==true)
    {
    if (ofr1==true){

  int          val          =          stmt.executeUpdate("INSERT          into          enrollment
VALUES('"+studentID+"','"+course1+"',1,'"+semester+"','"+year+"',null,null)");

  data[i++]=course1+" has been added";
            jList1.updateUI();

    }

    if (ofr2==true){

  int          val          =          stmt.executeUpdate("INSERT          into          enrollment
VALUES('"+studentID+"','"+course2+"',1,'"+semester+"','"+year+"',null,null)");

data[i++]=course2+" has been added";
            jList1.updateUI();
    }

    if (ofr3==true){

  int          val          =          stmt.executeUpdate("INSERT          into          enrollment
VALUES('"+studentID+"','"+course3+"',1,'"+semester+"','"+year+"',null,null)");

data[i++]=course3+" has been added";
            jList1.updateUI();
    }
    if (ofr4==true){

  int          val          =          stmt.executeUpdate("INSERT          into          enrollment
VALUES('"+studentID+"','"+course4+"',1,'"+semester+"','"+year+"',null,null)");

data[i++]=course4+" has been added";
            jList1.updateUI();
    }
    if (ofr5==true){

  int          val          =          stmt.executeUpdate("INSERT          into          enrollment
VALUES('"+studentID+"','"+course5+"',1,'"+semester+"','"+year+"',null,null)");
data[i++]=course5+" has been added";
            jList1.updateUI();
```

```
    }
        }
    }

    catch (SQLException ex) {
// handle any errors
        System.out.println("SQLException: " + ex.getMessage());
        System.out.println("SQLState: " + ex.getSQLState());
        System.out.println("VendorError: " + ex.getErrorCode());
    } finally {


        if (stmt != null) {
            try {
                stmt.close();
            }
            catch (SQLException sqlEx) {
            } // ignore
            stmt = null;
        }
    }

    }



    /**
    * @param args the command line arguments
    */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Receiver().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JList jList1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextField jTextField1;
```

```
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField5;
    private javax.swing.JTextField jTextField6;
    private javax.swing.JTextField jTextField7;
    private javax.swing.JTextField jTextField8;
    private javax.swing.JTextField jTextField9;
    // End of variables declaration

}
```